vb4arab
vb4arab.com

منتدى فيجوال بيسك للعرب



Microsoft .NET Framework

أعضاء منتدى فيجوال
بيسك للعرب

Microsoft®
.net™

بسم الله الرحمن الرحيم

* سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا إِنَّكَ أَنْتَ الْعَلِيمُ الْحَكِيمُ *

تقديم للسلسلة

أحببت أن أضع بعض المواضيع التي أجمعها من الشبكة العنكبوتية و من مستندات الMSDN تحت تصرف الإخوة الأعضاء، و أنظمها في سلسلة مقالات باللغة العربية تعالج مواضيع منوعة في ما يخص إطار العمل Microsoft .Net بطريقة مختصرة و مع أمثلة وافية وقد كان في اختياري للمواضيع نمط و منهجية بحيث أعالج المواضيع حسب المستويات مراعاة للإخوة المبتدئين و المحترفين على حد السواء، كما و نظمت المواضيع على مجموعة من الأقسام ستكون مخصصة: للمواضيع المتقدمة (هندسة وهيكلة التطبيقات)، الحيل و الأفكار البرمجية، التصميم، تحسين الأداء، والنظرات الخاطفة على بعض الفئات الجديدة أو المجهولة، وغيرها. كما أنها ستكون موجهة ل 3 مستويات I. للمبتدئين، II. للمتقدمين، III. للمحترفين.

أعلم أن هذا العمل ليس بالسهل أو الذي يمكن انجازه في وقت قصير، لكن سأحاول إن شاء الله تعالى أن أفي بوعدي و أضع مقالات بوتيرة منتظمة و أرجو من الإخوة الأعضاء أن يساعدوني بانجاز هذا العمل ليعم الخير، فالرجاء كل من يستطيع المساعدة أن يكتب في هذه السلسلة وأن يتقيد بالتصميم والنسق الذي ترونه حتى يكون العمل متقن و جميل و يعطي سمعة جيدة عن المنتدى و أعضائه و على الله تعالى الثواب و الأجر، وحتى يبقى للعرب مكان تحت الشمس.

وليد بوالطمين

كلية الهندسة/ معهد الإعلام الآلي و الرياضيات/ جامعة منتوري

مدينة قسنطينة – حاضرة العلم – الجزائر

الجمعة 28 جمادى الثانية من سنة 1428 للهجرة

الموافق ل 2007/07/14 م

تحسين سرعة الأداء بالاستعانة بالفئة Stopwatch



1. تمهيد

لا شك أن هدف أي مبرمج هو بلوغ درجة الاحتراف و بناء تطبيقات قوية و متينة من جميع النواحي، فالاعتناء بمظهر واجهة البرنامج و جعلها جذابة و سهلة الاستعمال للمستخدم أمر حيوي كما أن ملاحقة الأخطاء البرمجية (Debugging) و جعل التطبيق متين و خالي تماما من الأخطاء لا يقل أهمية عن الأول، لكن هل فكرت بالجانب الخاص بمصادر النظام و خاصة سرعة التنفيذ؟
إن من بين الأمور الأساسية التي تجعل البرنامج ينال رضا المستعمل هو سرعة التنفيذ، لذلك عليك الحرص على ذلك باختيار أسرع طرق التنفيذ من بين الخيارات المتوفرة لمعالجة مقطع من الكود. لكن كيف السبيل إلى المقارنة بين سرعات التنفيذ لمختلف الحلول المتوفرة؟
هذا السؤال سنجيب عليه إن شاء الله بين أسطر هذا المقال. لذلك شدوا الأحزمة سننطلق...

2. لمحة عن الفئة الجديدة Stopwatch

في سعيها الدءوب لتحسين و تطوير بيئة ال NET.، تضع لنا Microsoft ال 2.0 NET. Framework الذي يحتوي في مجمله على تحسينات و فئات جديدة جعلت منه غنيا مقارنة بسابقيه بحيث أصبح المبرمج يمتلك مجموعة جديدة كليا من الفئات و الأدوات التي جعلت من البرمجة متعة حقيقية.
من بين التحسينات التي أضافتها Microsoft على ال Net. Framework مجموعة جديدة من الفئات و من بينها الفئة Stopwatch التابعة لمجال الأسماء System.Diagnostics
رغم بساطتها إلا أنها ذات فائدة كبيرة للمبرمج فبواسطتها يمكنك احتساب وقت تنفيذ مقطع من الشيفرة بطريقة متناهية الدقة (دقة تصل إلى 10^{-7} جزء من الثانية) و من خلال 3 أسطر فقط!

2.أ. طريقة عمل الفئة

الفئة تمتلك مجموعة من الخصائص و الطرق تمكننا من حساب وقت التنفيذ وذلك بإتباع النهج التالي:
1 تهيئة الفئة باستعمال الطريقة Reset().
2 إعطاء الأمر ببدء احتساب الوقت باستعمال الطريقة Start() قبل بداية مقطع الكود المعني مباشرة.
3 إعطاء الأمر بتوقيف عملية الاحتساب باستعمال الطريقة Stop() بعد مقطع الكود.
4 معرفة الوقت المستنفذ من الخاصية Elapsed أو ElapsedMilliseconds أو ElapsedTicks.

2.ب. الطرق(Methods)

أحببت أن أختصر و أعرج على أهم الطرق التي نحتاجها.

الطريقة	الشرح
Reset()	تهيئة نسخة الفئة الحالية لعملية جديدة بحذف قيمة الوقت المحتسب سابقا (إعطاء القيمة 0 أو -null– Nothing) للخاصية ElapsedXXX) .
Start()	إعطاء الأمر ببداية الاحتساب. في حالة عدم تهيئة نسخة الفئة من قبل فإن عملية الاحتساب ستكون تراكمية أي أن القيمة الجديدة ستضاف إلى القيمة السابقة المخزنة في الخاصية ElapsedXXX.
Stop()	توقيف عملية الاحتساب و جعل الخاصية IsRunning تحمل القيمة false.

الجدول 1.2: بعض طرق الفئة Stopwatch.
ملاحظة: ElapsedXXX تشير إلى الخصائص ال Elapsed أو ElapsedTicks أو ElapsedMilliseconds

2.ت. الخصائص (Properties) و في عجالة

الخاصية	النوع	الشرح
Elapsed	System.TimeSpan	الوقت الكلي المحتسب بين الأمرين Start() و Stop() و ذلك على هيئة نسخة من الفئة TimeSpan.
ElapsedTicks	System.Int64	الوقت الكلي المحتسب بال Ticks (100 نانو ثانية 10^{-7} ثانية) = 1 Tick)
ElapsedMilliseconds	System.Int64	قيمة الوقت بالملي ثانية المحتسب من طرف النسخة الحالية.
IsRunning	System.Boolean	هل عملية الاحتساب قيد التشغيل أم لا.

الجدول 2.2: بعض خصائص الفئة Stopwatch

3. احتساب وقت تنفيذ مقطع من كود

سنستعمل مثال تطبيقي لشرح كيفية احتساب الوقت المستنفذ في تنفيذ مقطع من كود. و على سبيل المثال لا الحصر سنقوم بتعريف دالة تقوم بالإضافة المتكررة لحرف معين إلى متغير نوع System.String ن مرة و نحتسب الوقت اللازم لفعل ذلك.

3.أ. تحديد مجال الأسماء

كما قلنا، الفئة Stopwatch تابعة لمجال الأسماء System.Diagnostics

```
Imports System.Diagnostics 'VB .NET لمبرمجي ال
```

```
using System.Diagnostics; //C# لمبرمجي ال
```

3.ب. تعريف الدالة التي تقوم بعملية الإضافة المتكررة:

```
    Private Function Concat(ByRef Target As String, ByVal newChar As
Char, ByVal nb As Integer) As Boolean 'VB .NET لمبرمجي ال
        If nb <= 0 Then
            Return False
        Else
            Dim i As Integer
            For i = 1 To nb
                Target &= newChar
            Next
            Return True
        End If
    End Function
```

```
    //C# لمبرمجي ال
    private bool Concat (ref string Target,char newChar,int nb)
    {
            if (nb <= 0)        return false;
            else
            {
                    for (int i = 0; i < nb; i++)
                            Target += newChar;
                    return true;
            }
    }
```

الشيفرة 1.3: تعريف دالة الإضافة التراكمية.

3.ت. استعمال الفئة Stopwatch لحساب سرعة تنفيذ تكديس 50000 حرف في متغير سلسلة حروف.

```
    'VB .NET مبرمجي ال
    Private Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
        ' Stopwatch إنشاء نسخة من الفئة
        Dim mySWatch As New Stopwatch()
        Dim myStrTest As String
        ' ابدأ عملية الحساب
        mySWatch.Start()
        ' المقطع الذي نريد حساب وقت تنفيذه
        Concat(myStrTest, "A"c, 50000)
        ' توقيف عملية الحساب
        mySWatch.Stop()
        ' إظهار الوقت المستغرق لتنفيذ عملية إضافة 50000 حرف إلى المتغير
        Label1.Text = mySWatch.ElapsedMilliseconds.ToString & _
" ميلي ثانية"
    End Sub
```

```
//C# المبرمجي ال
private void button1_Click(object sender, System.EventArgs e)
{
      Stopwatch mySWatch = new Stopwatch();
      string    myStrTest;
      //ابدأ عملية الحساب
      mySWatch.Start();
      // المقطع الذي نريد حساب وقت تنفيذه
      Concat(ref myStrTest, 'A', 50000);
      //توقيف عملية الحساب
      mySWatch.Stop();
      // إظهار الوقت المستغرق لإضافة 50000 حرف إلى المتغير
      Label1.Text = mySWatch.ElapsedMilliseconds.ToString() + "  ميلي
ثانية"
}
```

الشيفرة 2.3: احتساب الوقت المستغرق لتكديس 50000 حرف.

4. بعض الأمثلة التطبيقية

سأعرض في هذه الفقرة مثالين عن بعض الحالات الشهيرة لاستعمال الفئة في مقارنة وقت تنفيذ مقطع يمكن كتابته بشكلين مختلفين، الأول في مقارنة بين الحلقتين التكراريتين For و For Each (for و foreach في ال C#) أما الثاني فسأعرض فيه مقارنة بين الفئتين String و StringBuilder و بلغتي ال Visual Basic .NET و Visual C#.

ملاحظات:

- كما تعلمون، فإن التنفيذ الأول للأمثلة (استدعاء الطرق لأول مرة من طرف المترجم) سيكون بطيئا جدا لأن كود الMSIL ستعاد ترجمته بواسطة المترجم JIT إلى لغة الآلة. لذلك بعد التنفيذ الثاني يمكننا حساب الوقت الحقيقي في الأمثلة.
- النتائج المتحصل عليها بعد التنفيذ تختلف من جهاز إلى آخر حسب مواصفاته فالنتائج المتحصل عليها عندي ليست بالضرورة ما ستتحصل عليه بعد تنفيذ الأمثلة (الاختلاف يكون في الأرقام و ليس في الحل الأحسن).
- الفئة StringBuilder تابعة لمجال الأسماء System.Text

4.أ. لمبرمجي ال Visual Basic .NET

4.أ.1. المثال الأول: مقارنة بين سرعة تنفيذ الحلقتين For و For Each

```
    Private Sub btnSWatchTest_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSWatchTest.Click
        Dim mySWatch As New Stopwatch()
        Dim myArrayList As New Collections.ArrayList()          'قائمة

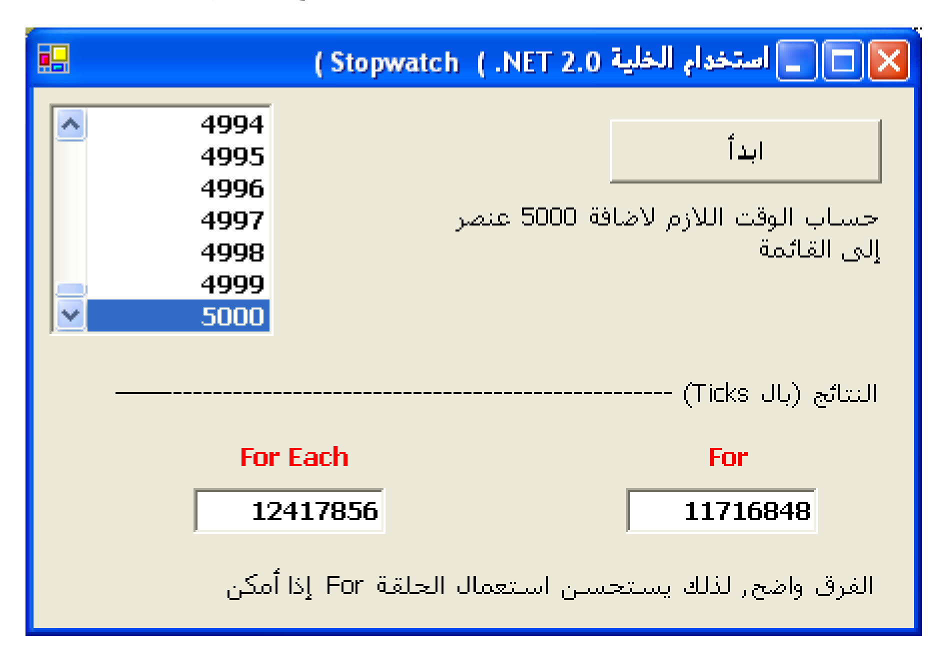
        Dim i As Integer
        'ملأ القائمة
        For i = 1 To 5000
            myArrayList.Add(i)
        Next

        'For Each حساب الوقت اللازم لتنفيذ المقطع باستعمال الحلقة التكرارية
        mySWatch.Start()
        For Each i In myArrayList
            lstItems.Items.Add(i)
        Next
        mySWatch.Stop()
        '10^-7 دقة الحساب
        txtForEach.Text = mySWatch.ElapsedTicks.ToString()

        'For حساب الوقت اللازم لتنفيذ المقطع باستعمال الحلقة التكرارية
        lstItems.Items.Clear()
        mySWatch.Reset()

        mySWatch.Start()
        For i = 0 To myArrayList.Count - 1
            lstItems.Items.Add(myArrayList(i))
        Next
        mySWatch.Stop()
        '10^-7 دقة الحساب
        txtFor.Text = mySWatch.ElapsedTicks.ToString()
    End Sub
```

الشيفرة 1.4: مقارنة وقت التنفيذ المستغرق باستعمال حلقتين مختلفتين مع القوائم.



الصورة 1.4: منظر النتائج و يبدو أن الحلقة For أحسن من نظيرتها For Each مع القوائم.

2.أ.4 المثال الثاني: مقارنة بين الخليتين String و StringBuilder

```
    Private Sub btnSWatchTest_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSWatchTest.Click
        Dim strTest As String
        Dim mySWatch As New Stopwatch()
        'System.String استعمال الفئة العادية
        mySWatch.Start()
        Call ConcatStr(strTest, "a"c, 50000)
        mySWatch.Stop()
        '10-7 دقة الحساب
        txtStr.Text = mySWatch.ElapsedTicks.ToString()

        'System.Text.StringBuilder استعمال الفئة
        mySWatch.Reset()
        mySWatch.Start()
        Call ConcatStrBuilder(strTest, "a"c, 50000)
        mySWatch.Stop()
        '10-7 دقة الحساب
        txtStrBuilder.Text = mySWatch.ElapsedTicks.ToString()
    End Sub
    Private Function ConcatStr(ByRef Target As String, ByVal newChar As Char,
ByVal nb As Integer) As Boolean
        If nb <= 0 Then
            Return False
        Else
            Dim i As Integer
            For i = 1 To nb
                Target &= newChar
            Next
            Return True
        End If
    End Function
    Private Function ConcatStrBuilder(ByRef Target As String, ByVal newChar As
Char, ByVal nb As Integer) As Boolean
        If nb <= 0 Then
            Return False
        Else
            Dim strBuilder As New System.Text.StringBuilder(nb، nb)
            strBuilder.Append(newChar, nb)
            Target = strBuilder.ToString()
            Return True
        End If
    End Function
```

الشيفرة 2.4: مقارنة وقت التنفيذ المستغرق لتكديس 50000 حرف باستعمال الفئتين.



الصورة 2.4: منظر النتائج و تظهر القوة الرهيبة للفئة StringBuilder

4.ب. لمبرمجي ال Visual C#

4.ب.1. المثال الأول: مقارنة بين سرعة تنفيذ الحلقتين for و foreach

```
private void btnSWatchTest_Click(object sender, System.EventArgs e)
{
  Stopwatch mySWatch = new Stopwatch();
  Collections.ArrayList myArrayList = new Collections.ArrayList();
  //ملأ القائمة
  int i=1;
  for (; i <= 5000; i++)
      myArrayList.Add(i);

  //foreach حساب الوقت اللازم لتنفيذ المقطع باستعمال الحلقة التكرارية
  mySWatch.Start();
  foreach (int i in myArrayList)
      lstItems.Items.Add(i);
  mySWatch.Stop();

  '10-7 دقة الحساب
  txtForEach.Text = mySWatch.ElapsedTicks.ToString();

  //for حساب الوقت اللازم لتنفيذ المقطع باستعمال الحلقة التكرارية
  lstItems.Items.Clear();
  mySWatch.Reset();

  mySWatch.Start();
  for (i = 0; i <= myArrayList.Count - 1;i++)
      lstItems.Items.Add(myArrayList(i));
  mySWatch.Stop();

  '10-7 دقة الحساب
  txtFor.Text = mySWatch.ElapsedTicks.ToString();
}
```

الشيفرة 3.4: إعادة المثال 4.أ.1. بلغة ال C# .

4.ب.2. المثال الثاني: مقارنة بين الخليتين String و StringBuilder

```
private void btnSWatchTest_Click(object sender, System.EventArgs e)
{
  string strTest ;
  Stopwatch mySWatch =new Stopwatch();

  //System.String استعمال الفئة العادية
  mySWatch.Start();
  ConcatStr(strTest, 'a', 50000);
  mySWatch.Stop();

  //10⁻⁷ دقة الحساب
  txtStr.Text = mySWatch.ElapsedTicks.ToString();

  mySWatch.Reset();

  //System.Text.StringBuilder استعمال الفئة
  mySWatch.Start();
  ConcatStrBuilder(strTest, 'a', 50000);
  mySWatch.Stop();

  //10⁻⁷ دقة الحساب
  txtStrBuilder.Text = mySWatch.ElapsedTicks.ToString();
}
private bool ConcatStr(ref string Target ,char newChar , int nb )
{
      if (nb<=0)
            return false;
      else
      {
            for (int i = 0; i < nb; i++)
                  Target += newChar;
            return true;
      }
}
private bool ConcatStrBuilder(ref string Target ,char newChar , int nb )
{
      if (nb <=0 )
            return false;
      else
      {
            System.Text.StringBuilder strBuilder = new
System.Text.StringBuilder(nb,nb);
            strBuilder.Append(newChar,nb);
            Target = strBuilder.ToString();
            return true;
      }
}
```

الشيفرة 3.4: إعادة المثال 4.أ.2. بلغة ال C# .

5. ماذا عن المبرمجين غير المواكبين لتكنولوجيا الدوت نت ؟!!!

يمكن الاستعانة بدوال ال API الخاصة بالويندوز بالنسبة لمبرمجي ال Visual Basic و ال Visual C++ مثلا.

و على سبيل المثال لا الحصر، يمكن استخدام الدالة GetTickCount التي تعطينا الوقت المستغرق ب ال Ticks (100 نانوثانية = 1 Tick) منذ بداية Session الويندوز.

```
'vb6.0 لمبرجي ال
Private Declare Function GetTickCount Lib "kernel32" () As Long
```

```
/*
      C++ لمبرجي ال
      windows.h :header الدالة معرفة في ال
*/
#include "windows.h"
```

و لاستعمالها، يجب العمل كالتالي:

1 تخزين القيمة المرجعة من الدالة عند البداية في متغير من نوع long.

2 تخزين القيمة المرجعة من الدالة عند النهاية في متغير من نوع long.

3 حساب الفرق بين قيمتي المتغيرين و الذي يمثل الوقت المستغرق لتنفيذ المقطع (بالتقريب لأن زمن استدعاء الدالة يحتسب أيضا).

6. خاتمة

بسيطة، سريعة التعلم، جد مفيدة، هذا ما يمكن قوله عن الفئة الجديدة Stopwatch التي تعتبر واحدة من التحسينات العملية في نسخة إطار العمل .NET Framework 2.0 و لا يجب تجاهلها لأنها أداة جيدة للمبرمجين الجادين.

أخيرا عليك أخي المبرمج الأخذ في عين الاعتبار أن الوصول إلى أسرع الحلول أمر مهم في جعل برنامجك اقتصادي في مصادر النظام، فإهمال جانب كهذا لا يمت بصلة إلى الحرفية و المهنية في التطوير البرمجي.

وعموما تدخل مرحلة تسريع الأداء في نطاق التحسينات على النسخة الأصلية حيث لا يمكن أن تقوم بتجريب جميع الإمكانيات و المقارنة بينها مع بداية تطوير النسخة الأولى لذلك عليك التركيز أولا على الهيكل وبعدها تتحول إلى التحديثات.

